DECUS
PROGRAM LIBRARY

DECUS NO.	8-393
TITLE	QUEING TCØ1/TU55 DECTAPE ROUTINES
AUTHOR	James Crapuchettes
COMPANY	Stanford Electronic Labs Stanford, California
DATE	September 29, 1970
SOURCE LANGUAGE	PAL III

QUEING TCØ1/TU55 DECTAPE ROUTINES

DECUS Program Library Write-up DECUS NO. 8-393

New TC01/TU55 DECtape Routines:

These routines, which are a much modified version of DEC-08-FUBO (previously DEC-08-31U), provide the user with the ability to read and write 128 words (one memory page) from/onto standard 129 word DECtape blocks. Successive blocks are transferred into/from successive 128 word areas of memory. The routines will transfer into/from any memory field, will begin searching in either the forward or the reverse direction for the DECtape block at which the transfer will begin, and will queue one read/write request to keep the DECtape in motion (and transferring data) as continuously as possible.

The only restrictions placed on these routines is that they must be located in field 0 of memory and that they will read and write in the forward direction only. Also, as with other DECtape routines, they cannot transfer data across field boundaries, i.e. location 7777 of a field is followed by location 0 of the same field.

Areas of application for these routines:

These routines can be used to advantage in any program which processes data which is read from or written onto DECtape at a higher rate than the average transfer rate of DECtape transfer. The processing rate, when compared to the maximum DECtape transfer rate of about 7.5 Kwps, falls into three categories:

1. Whenever the processing rate is always higher than the transfer rate, these routines can keep the DECtape in continuous motion, causing the actual transfer rate to approach the maximum rate, and allowing the data on one complete DECtape to be transferred and processed in about 1 minute. This can be particularly important when only a small area of memory is available for DECtape buffers, because other routines would then need a considerable amount of time to search for the start block for the transfer, decreasing the average transfer rate by as much as a factor of 13.

2. Whenever the processing rate is lower than the maximum rate of the DECtape but greater than the average rate (which includes the time spent searching for the transfer start block), using these routines in conjunction with multiple (more than 2) buffers can raise the average rate (compared to the average rate of other routines) as much as is needed for maximum efficiency in the program. Note that more than 2 buffers are needed in order that there be sufficient buffer space after one buffer is filled, for the routines to search for the transfer start block in case the DECtapes have stopped. Since these routines will queue a request, the actual size of the buffers is not necessarily important and several small buffers may work better than 1 or 2 large buffers.

3. Whenever the processing rate is lower than the average transfer rate of the DECtape (when using other routines) but may include high peak rates, the technique of using several small buffers may be used to advantage as described above.

The disadvantages of using these routines, when compared to other DECtape routines in usage, is that they do not attempt to re-try a read or write when an error occurs and that they require about 1 1/4 pages of core. The last disadvantage is somewhat lessened by the ability of the routines to handle small buffers.

Calling sequences:

These routines consist of several subroutines. Of these, two (RW128 and DWAIT) have direct use in user programs and two others (DTRAR and DTXAR) may have usage in special instances.

Subroutine RW128:

This subroutine is the Read/Write subroutine which controls DECtape transfers. It has a calling sequence:

```
JMS (RE128)     /Effective JMS to RW128
CORE            /Start address in memory for the transfer
USFW            /Unit, Search, Field and Write (see below)
BLKNO           /DECtape block number at which transfer starts
-BLKS           / -(Number of DECtape blocks to transfer)
Return here with DECtape in motion
```

The control word, USFW, is divided into the following fields:

```
U (Bits 0-2) = DECtape unit number, 0 (for 8) through 7
S (bit 3)    = Search start direction.  0 to start in reverse,
               1 to start forward.  (Xfer always forward.)
F (Bits 6-8) = Field of memory for the data transfer.
W (Bit 9)    = Write or read.  1 for write, 0 for read.
```

The search direction bit, S, allows the program to specify the direction in which the DECtape should begin moving when it is searching for BLKNO. This can save some time when the desired direction is known.

When program control returns from the subroutine, a DECtape will be in motion and the AC will be cleared (= 0). The state of the Link is uncertain. The DECtape unit in motion at the time that program control returns may or may not be the same unit just requested. Furthermore, the unit may be performing one of the following three functions: a) searching for BLKNO on the unit just specified, b) transferring data as specified by the last request made to the program, or c) transferring data as specified by this request. Due to this fact, a small amount of care must be exercised when using these routines, as will be described in the section on usage.

Subroutine DWAIT:

This subroutine is used by a calling program (and also by RW128) to find the current status of the DECtape and RW128 queue. Control will not return will not return to the calling program until one of two conditions is true. The desired condition is determined by the setting of the Link when the subroutine is called (the AC should be cleared). If the Link is set (a 1), control will return when (if) no DECtape is in motion. If the Link is clear (a 0), control will return when (if) either no DECtape is in motion or the queue in RW128 is empty. This subroutine has a calling sequence:

```
STL or CLL      /SeT Link or CLear Link
JMS (DWAIT)     /Effective JMS to DWAIT
Return here when specified condition true (AC = 0)
```

Subroutine DTRAR:

This subroutine checks the motion bit in the DECtape A (command) register and returns to the calling address+1 if the bit is a 0 (no DECtape in motion) or to the calling address+2 if the bit is a 1 (a DECtape is in motion). It has a calling sequence:

```
JMS (DTRAR)     /Effective JMS to DTRAR, AC = 0
Return here if no DECtape in motion (AC = 0)
Return here if a DECtape is in motion (AC = 0)
```

This subroutine should be called with the AC cleared and returns with the AC cleared and the Link undisturbed. It can be used by a program to determine the status of the DECtape control.

Subroutine DTXAR:

This subroutine reads the DECtape A (command) register, masks the AC (by an AND) with the contents of the calling address+1, issues the DTXA iot command (which exclusive-ors the contents of the AC with the contents of the A register), and then returns to the calling address+2 with the AC clear. Any bits set in the AC when the subroutine is called will be inclusive or'ed with the contents of the DECtape A register (the read) before masking. This subroutine has a calling sequence:

```
JMS (DTXAR)     /Effective JMS to DTXAR
MASK            /Mask for AC or'ed with A register.
Return here with AC cleared.
```

This subroutine, with a mask of 200 (octal), will stop any DECtape unit which is in motion. This can be used in a program when something occurs which requires that the DECtape stop immediately. Any request left in the queue in RW128 will be ignored after this is done.

Usage of the routines:

Usage of this routine is somewhat similar to that described for its parent in the DEC writeup. If the user is unfamiliar with this routine, it is suggested that he read the writeup available from DEC.

This routine requires three symbols to be defined on page 0 of memory (field 0). These three symbols are MCOM, DIS, and DTERR. MCOM is used by the DECtape interrupt handling routines to point to the current entry point in the interrupt handler. It must be defined by the user in his program but is initialized by the routine. Its usage by the user is described in the DEC writeup. DIS must be a pointer to the user's interrupt dismiss routine. This is also described in the DEC writeup. DTERR must be a pointer to the user's DECtape error processing routine. When a DECtape error occurs, the routines stop the DECtape, load the AC with the contents of the DECtape B (status) register, and then execute a JMP I DTERR. Unlike the DEC version, the status (errors) will be undisturbed in the B register also. A few more comments on error processing will follow shortly.

Unlike the DEC routines, this routine does NOT include an ION instruction to enable the program interrupt. It is important that the user program enable the interrupt shortly after the first time that the routine is called. If it is a problem that the ION is not included, there is room to add this one instruction in the subroutine labeled DSSET, which sets up a search for BLKNO.

In some programs it may be necessary to process a DECtape error with the program interrupt on (as in the case of buffered Teletype

output). If it is desired to do this, it is necessary to make a small change in the interrupt skip chain to properly ignore an illegal DECtape flag. This illegal flag is caused by the DTXA instruction which stops the DECtape unit. In the case of a select error (which is caused by either selecting an off-line unit or by trying to write on a write-locked unit), this DTXA instruction will cause the error to occur again. Simply checking for a DECtape flag (with a DTSF) to determine the device which caused an interrupt will cause the program to go to the DECtape interrupt handler with possible disastrous results. To solve this problem, the DTXA instruction which stops the unit also clears the bit in the DECtape A register which allows the DECtape flag to cause a program interrupt. The interrupt skip chain should therefore check this bit before going to the DECtape handler. An example follows:

```
/Save the AC and Link after the interrupt...
DTSF            /Is the DECtape flag set?
JMP .+5         /No, try the next device
DTRA            /Yes, is control connected to P.I.?
AND P4          / C(P4) = 4
SZA CLA         /If AC = 0, not connected, try next device
JMP I MCOM      /All OK, go handle DECtape interrupt
... /Continue with device identification routine
```

This modified routine will properly handle "true" DECtape interrupts while ignoring the flag if it could not have caused the interrupt.

Because RW128 will queue a DECtape read/write request, care must be used to keep from using a buffer before the transfer into or out of it is finished. The table on the following page shows the possible functions that can be in progress after repeated calls to RW128 and to DWAIT. It is suggested that the user read and attempt to understand the

table before using the routine. The following words are used in the table:

SEARCH	Signifies that the routine is searching the DECtape for BLKNO.
XFER	Signifies that a data transfer (read/write) is taking place.
Q	Signifies that the Queue in RW128 (which has a depth of one request) is full. If it is necessary for RW128 to perform a SEARCH for a specified request, the Queue will remain full until the actual XFER begins for that request.
$\overline{Q}$	Signifies that the Queue in RW128 is empty.
n-1, n, n+1,	Signify the number of the current request. In each of the examples shown, the current request is number n, the last request is n-1, etc.

Should there be any questions about the routine or its use, I will do what I can to be of help.

Jim Crapuchettes
Stanford Electronics Labs
Stanford, Calif.

Use of RW128: "(wait)" means that program control did not return from the routine immediately

Function		Status of DECtape & Queue	
before call →	STOP, $\overline{Q}$	SEARCH, Q, n-1	XFER, $\overline{Q}$, n-1
JMS RW128 . (n) . . .		(wait)	
on return → CLL JMS DWAIT	SEARCH, Q, n	XFER, Q, n-1	XFER, Q, n-1
on return → (note that status is same in all cases.)	XFER, $\overline{Q}$, n	XFER, $\overline{Q}$, n	XFER, $\overline{Q}$, n

---

before call →	STOP, $\overline{Q}$	SEARCH, Q, n-1	XFER, $\overline{Q}$, n-1
JMS RW128 . (n) . . .		(wait)	
on return → . . .	SEARCH, Q, n	XFER, Q, n-1	XFER, Q, n-1
JMS RW128 . (n+1) . . .	(wait)	(wait)	(wait)
on return → . . .	XFER, Q, n	XFER, Q, n	XFER, Q, n
JMS RW128 . (n+2) . . .			
on return →	XFER, Q, n+1	XFER, Q, n+1	XFER, Q, n+1

RW128, QUEING DECTAPE ROUTINES: SOURCE, BIN & LISTING 15 OCT 1970

NAME	BLK#	BLKS	S.A.	CORE LOCS	
INDEX	7777	8	7777	07400-07577	
START	0007	9	7667	07400-07577	
UPDATE	0020	7	7600	06000-07577	
CHANGE	0027	5	7600	06400-07577	
GETSYS	0034	5	7600	06200-07377	
XSYS	0041	13	7667	04400-07577	
XEDIT	0056	11	0147	00000-02577	
XPAL	0071	31	0200	00000-07577	
XLOAD	0130	5	7200	06400-07577	
XMOVE	0135	11	0200	00000-02577	
XINDEX	0150	9	0200	00000-02177	
XODT8	0161	3	7000	07000-07577	
XLIST	0164	12	0200	00000-02777	
XDUP	0200	4	0200	00000-00777	
XTAPE	0204	24	0177	00000-05777	
DECU31	0234	12	7667	07400-07577	
SRWTAP	0250	20	7667	07400-07577	
CONVT	0274	31	0200	00000-07577	
SRW128	0333	30	7667	07400-07577	Source file
BRW128	0371	6	7667	07400-07577	Binary file
LRW128	0377	50	7667	07400-07577	Listing file
RW128T	0461	3	7667	00000-00577	Test program

NUMBER OF FILES: 21
NEXT FREE BLOCK: 0464
NEXT FREE WORD: 6236
    WORDS LEFT: 0341